Considerations for Algorithm Selection and C Programming Style for the SRC-6E Reconfigurable Computer

Russ Duren and ***Douglas Fouts***
Naval Postgraduate School

Abstract: The architecture and programming environment of the SRC-6E Reconfigurable Computer was presented in the 2002 MAPLD International Conference [1]. That paper described how the programmer could trade off chip area for execution speed. In this paper we discuss additional aspects of programming the SRC-6E. In particular we examine classes of algorithms that best benefit from the SRC-6E's architecture. Additionally, we examine how the SRC-6E compiler interprets various C language constructs.

Introduction

The SRC-6E reconfigurable computer provides a state of the art platform for developing and executing programs that take advantage of multi-million gate Xilinx FPGAs combined with Intel Pentium Processors. The architecture of the SRC-6E has been designed so that it can be rapidly modified to leverage advances in both personal computers and FPGAs. The programming environment of the SRC-6E allows programmers to utilize the resulting computing power with minimal consideration of the hardware itself [2].

The current version of the SRC-6E allows the programmer to access two Xilinx Virtex II XC2V6000TM FPGAs. The two FPGAs share 24 MB of RAM referred to as onboard memory or OBM. The Pentium processors and FPGAs communicate over a 64-bit interface with an 800 MB/s peak bandwidth. This bus can be used for message passing and for DMA transfers between the PC's memory and OBM. Each FPGA is connected to the OBM with six buses that provide an aggregate peak bandwidth of 4800 MB/s. The two FPGAs are interconnected by three buses with an aggregate peak bandwidth of 2400 MB/s. Additional interface buses are provided to connect each FPGA to off board data sources and sinks. The latter interfaces can be used for real-time embedded processing.

The architecture of the SRC-6E allows programmers to realize significant speed improvements when executing certain classes of algorithms. The speed improvements result from the use of the resources provided by the FPGAs. In practice, the speedup depends greatly on the properties of the algorithm. Certain algorithms are better suited for implementation in an FPGA. Such algorithms will benefit most from the SRC-6E. Although the SRC-6E provides very high bandwidth interconnections, certain algorithms can saturate these resources limiting performance gains. This paper examines these issues.

The SRC-6E compiler supports programming in two high order languages (HOL). Programs are developed and debugged in C or FORTRAN in a Linux environment. The intent is to support programming entirely in the HOL. When performance demands additional speed, the SRC-6E compiler allows the user to develop highly optimized macros using Verilog, VHDL or Schematic capture. These macros are accessed by function calls within the HOL source program. From the programmer's point of view this is similar to calling assembly language routines for increased performance on a standard microprocessor.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **20 AUG 2004**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **Considerations for Algorithm Selection and C Programming Style for the SRC-6E Reconfigurable Computer**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001694, HPEC-6-Vol 1 ESC-TR-2003-081; High Performance Embedded Computing (HPEC) Workshop (7th)., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**20**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

The SRC-6E compiler allows the programmer to specify portions of code to be executed on either FPGA. The compiler translates the HOL user code to highly pipelined operations on the FPGAs. Pipelining allows complicated HOL routines to produce results every 10 nanoseconds. Additional speed increases may be obtained through the use of parallel computations. To a large extent, the SRC-6E compiler frees the programmer from having to think about the target hardware when developing code. Still, certain HOL constructs can lead to vastly different performance. This paper addresses this further.

RELATED WORK

FPGAs and ASICs have been used for high performance digital signal processing for many years. Parhi discusses VLSI design techniques that trade off speed, area, and power dissipation [3]. Chodowiec discusses performance gains made possible by reconfigurable computing [4]. He examines gains and limitations obtained from pipelining, parallel circuitry, and loop unrolling. Peterson and Drager list a number of reconfigurable computing platforms and discus their application to defense-related processing.

REFERENCES:

1. Fidanci, O.D., Diab, H., El-Ghazawi, T., Gaj, K. and Alexandridis, N. "Implementation trade-offs of Triple DES in the SRC-6e Reconfigurable Computing Environment," 5th MAPLD International Conference, Laurel, Maryland, September, 2002.

2. SRC-6E C Programming Environment v1.3 Guide, SRC Computers, Inc. April 11, 2003

3. Parhi, K.K., "VLSI Digital Signal Processing Systems, Design and Implementation," John Wiley & Sons, NY, 1999.

4. Chodowiec, P.R., "Comparison of the Hardware Performance of the AES Candidates Using Reconfigurable Hardware," Master's Thesis, George Mason University, Spring 2002.

5. Peterson, G.D. and Drager, S.L., "Accelerating Defense Applications Using High Performance Reconfigurable Computing," GOMAC, 2003.

NAVAL
POSTGRADUATE
SCHOOL

Baylor University
Engineering & Computer Science

Algorithm and Programming Considerations for Embedded Reconfigurable Computers

Russell Duren,
Associate Professor
Engineering And
Computer Science
Baylor University
Waco, Texas

Douglas Fouts, Professor
Daniel Zulaica, Engineer
Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California

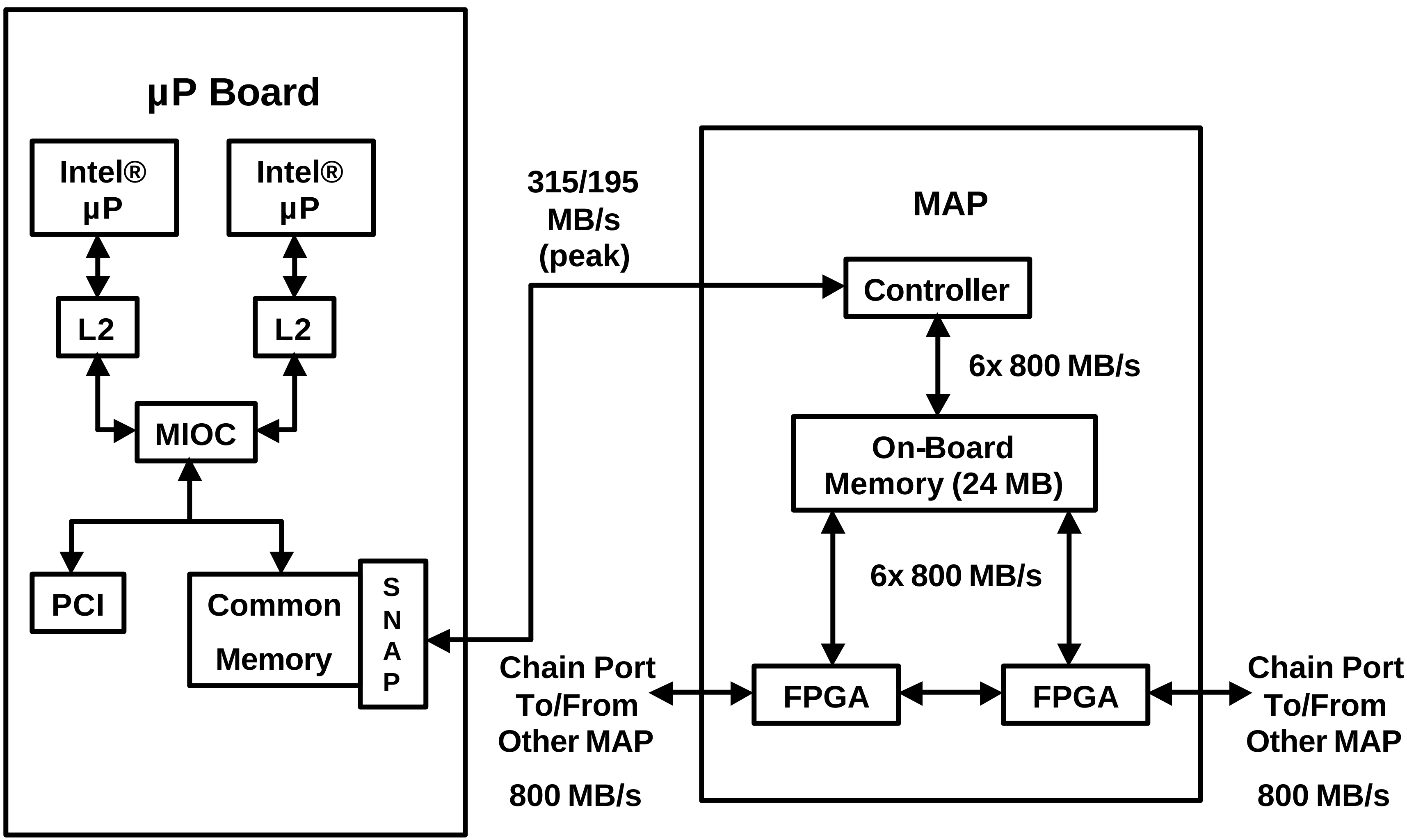
OBJECTIVES

- Experience and evaluate the learning curve required to become proficient at developing software for the SRC-6e reconfigurable computer.
- Benchmark and compare the performance and correctness of the SRC-6e against computers with traditional architectures.
- Establish libraries of open-source functions.
- Develop a hardware interface between the SRC-6e and a U.S. Navy AN/SPS-65V search radar.
- Develop software for real time processing of radar signals on the SRC-6e.
- Study and experiment with programming languages, methodologies, and environments to move reconfigurable computing closer to the environment most applications developers are already familiar with.

SRC-6e RECONFIGURABLE COMPUTER

- LINUX CLUSTER OF TWO PCs
- EACH PC HAS
  - TWO 1000 MHZ INTEL XEON® PROCESSORS
  - COMMON MEMORY
  - SNAP PORT TO MULTI ADAPTIVE PROCESSOR (MAP)
- EACH MAP HAS
  - TWO USER-PROGRAMMABLE XILINX VIRTEX-II FPGAs (6 M GATES EACH)
  - ONE XILINX VIRTEX-II CONTROL FPGA (NOT USER PROGRAMMABLE)
  - ON-BOARD MEMORY
  - SNAP PORT TO PC
  - TWO 96-BIT WIDE CHAIN PORTS TO OTHER MAP
- PROGRAMS WRITTEN IN C OR FORTRAN
  - USER IDENTIFIES WHICH PART(S) OF PROGRAM ARE CONVERTED TO FPGA CIRCUITRY FOR (HOPEFULLY) INCREASED EXECUTION SPEED
  - FPGA CODE CAN ALSO BE WRITTEN IN VHDL OR VERILOG
  - FPGA CAN ALSO BE PROGRAMMED SCHEMATICALLY OR WITH IP CORES

SRC-6E ARCHITECTURE (HALF)



MAP SOFTWARE DEVELOPMENT

- CODE FOR FPGAs IS ISOLATED IN EXTERNAL FUNCTION
- SRC COMPILER TRANSLATES C SOURCE CODE INTO VERILOG
- VERILOG IS COMPILED TO FPGA CIRCUITRY
- MAP CAN ALSO BE PROGRAMMED WITH VERILOG, VHDL, IP CORES, OR SCHEMATICALLY
- FPGA CIRCUITRY DEEPLY PIPELINED WITH 100 MHZ CLOCK (10 NS)
- LARGE PIPELINE FILL TIME (LARGE LATENCY)
- CALLS ARE INSERTED IN THE MAIN PROGRAM TO
  - INITIALIZE THE MAP
  - TRANSFER INPUT DATA FROM COMMON MEMORY TO ON-BOARD MEMORY
  - CALL THE EXTERNAL FUNCTION
  - TRANSFER OUTPUT DATA FROM ON-BOARD MEMORY TO COMMON MEMORY
  - RELEASE THE MAP (OPTIONAL)

LIMITATIONS OF MAP C COMPILER

- LIMITED SUBSET OF C LANGUAGE
- LIMITED DATA TYPES (32-BIT AND 64-BIT INTEGERS)
- SNAP PORT DATA TRANSFERS ARE ALWAYS 32-BIT WORDS AND MUST BE ALIGNED ON A 4-BYTE BOUNDARY
- DATA ARRAYS MUST BE ALIGNED ON A 4-BYTE BOUNDARY
- SIX ON-BOARD MEMORY MODULES
- ONE ARRAY PER MEMORY MODULE
- ONE READ OPERATION OR ONE WRITE OPERATION PER MEMORY MODULE PER CLOCK
- NO SUPPORT FOR ACCESS TO CHAIN PORTS

- FUTURE VERSIONS OF MAP C COMPILER WILL FIX SOME, BUT NOT ALL, OF LIMITATIONS

EVALUATING THE SRC-6e AND MAP COMPILER

- EASE OF USE (SOMEWHAT SUBJECTIVE)
  - ALGORITHM/CODE MODIFICATIONS REQUIRED TO MAKE USE OF MAP
  - LIMITATIONS IMPOSED BY COMPILER
  - LIMITATIONS IMPOSED BY HARDWARE
- TYPICAL MEASURES OF COMPILER PERFORMANCE
  - CODE EXECUTION SPEED
  - OBJECT CODE SIZE (TRANSLATES TO FPGA CIRCUIT AREA)

- THE CORDIC ALGORITHM
  - USED TO EXTRACT PHASE INFORMATION FROM AN I/Q-ENCODED (IN-PHASE/QUADRATURE) RADAR SIGNAL
  - USES SUCCESSIVE APPROXIMATION TECHNIQUE TO ESTIMATE ARCTANGENT FUNCTION
  - NUMBER OF ITERATIONS DETERMINES ACCURACY
  - EASILY PIPELINED
  - USES SHIFT AND ADD ALGORITHM, NO MULITPLICATION OR DIVISION

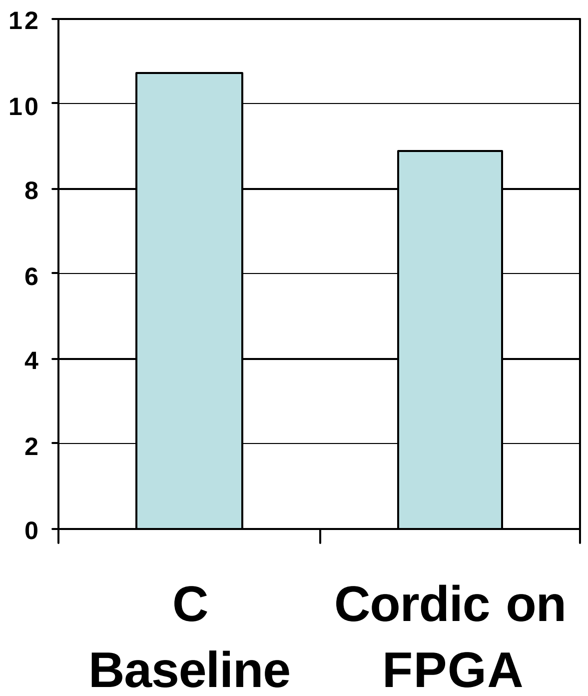
CORDIC TEST CASES

- C VERSION USING ONLY INTEL CPUs
- C VERSION USING SRC C COMPILER FOR MAP
  - 32-BIT INTEGER VARIABLES
  - 11 ITERATIONS
  - MOST CLOSELY MATCHES IP CORE NUMBER 2
- THREE VERSIONS OF MAP CODE GENERATED WITH XILINX IP CORE GENERATOR TOOL
- COMMON “MAIN” PROGRAM
  - CALLS CORDIC ROUTINE 256 TIMES
  - EACH CALL CALCULATES 249984 ARCTANGENTS
- DATA INPUTS TO CORES ARE 8 BITS WIDE
- INTERFACE TO CALLING ROUTINE USES 32-BIT INTEGERS

CORDIC EXECUTION TIME

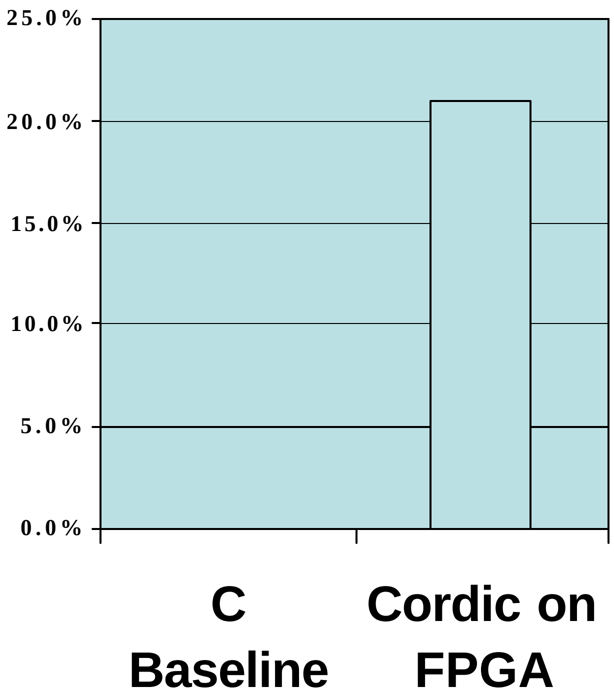
1000 MHZ INTEL XEON VERSUS 1000 MHZ XEON + MAP

EXECUTION TIME



20% DECREASE

FPGA SPACE USAGE

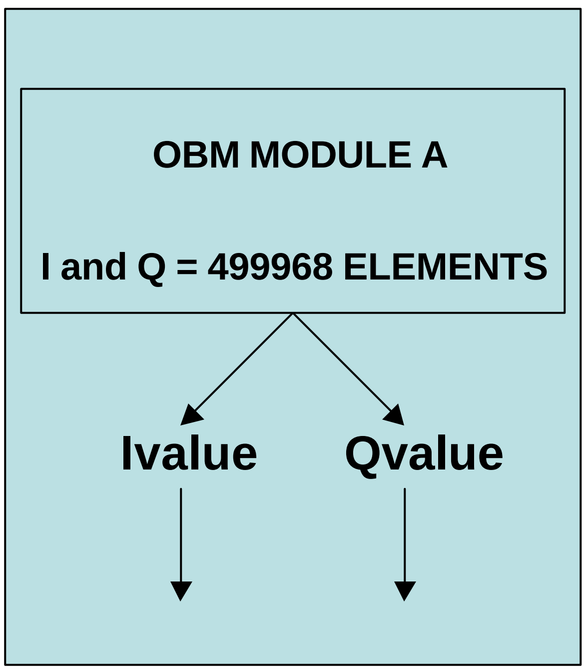


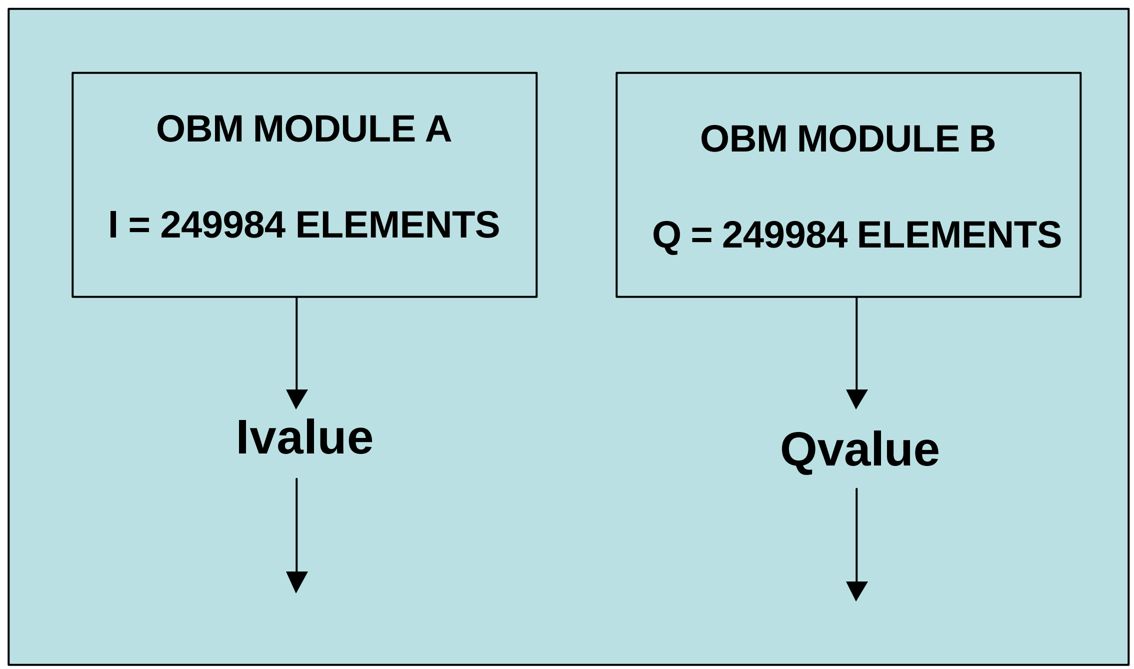
NAVAL POSTGRADUATE SCHOOL

Baylor University Engineering & Computer Science

REDUCING MAP DATA ACCESS TIME

- ORIGINAL MAP CODE STORES I AND Q INPUT DATA IN SINGLE ARRAY
- ARRAYS ALLOCATED TO MODULES IN ON-BOARD MEMORY, BUT ONLY ONE READ OR ONE WRITE OPERATION CAN BE PERFORMED ON EACH CLOCK CYCLE WITH EACH MEMORY MODULE
- USE OF 2 ARRAYS ALLOCATED TO 2 MEMORY MODULES, ONE FOR I AND ONE FOR Q INPUT DATA, REDUCES DATA ACCESS TIME

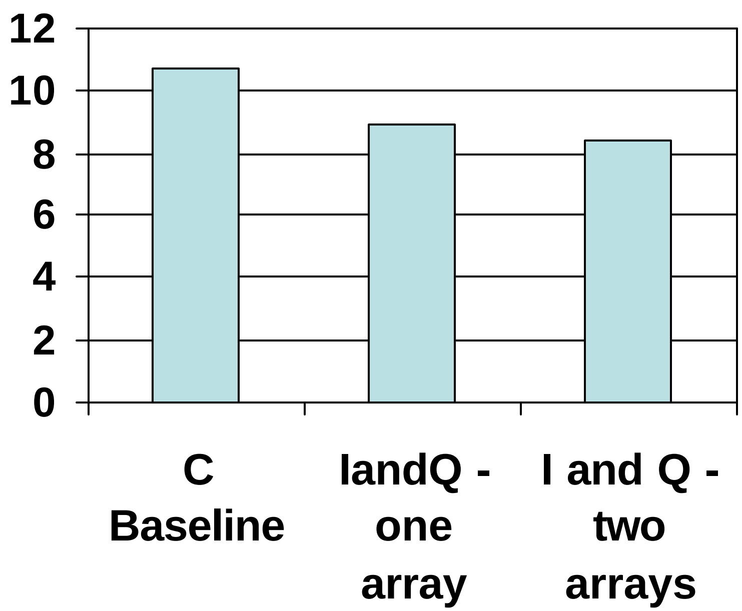




TEST RESULTS

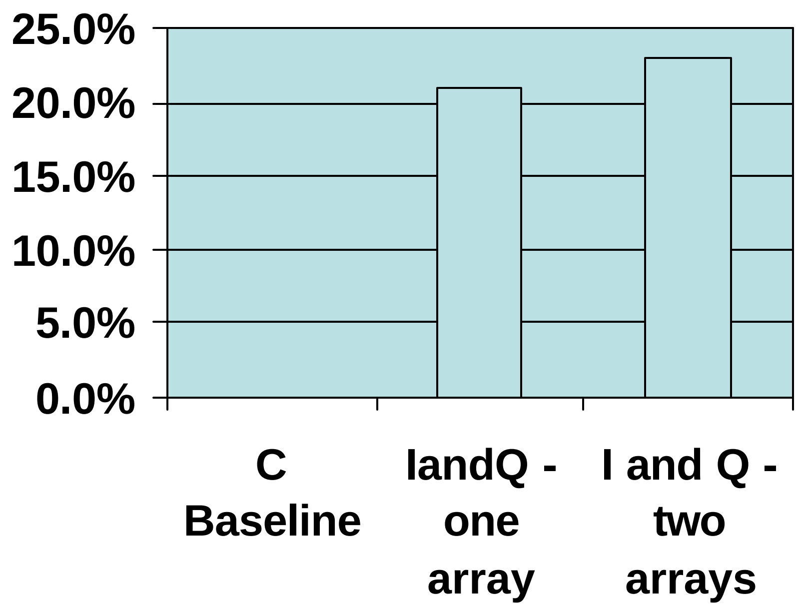
USING SEPARATE ARRAYS TO STORE I AND Q IMPROVES CORDIC EXECUTION TIME AT THE COST OF MORE FPGA SPACE USAGE

EXECUTION TIME



5% DECREASE

FPGA SPACE USAGE



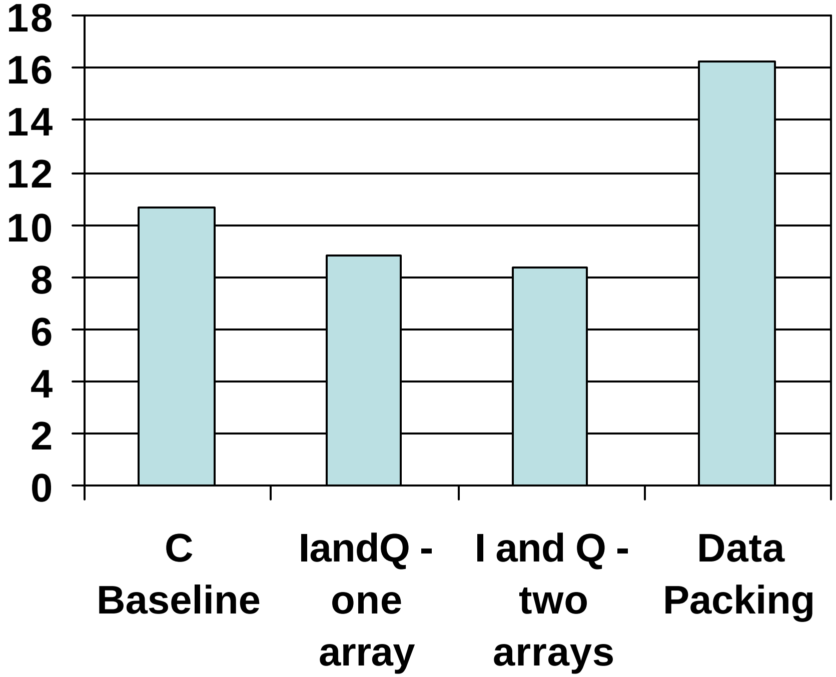
9% INCREASE

REDUCING MEMORY DATA TRANSFER TIME

- ORIGINAL MAP CODED ALLOCATED ONE 8-BIT I VALUE AND ONE 8-BIT Q VALUE PER 4-BYTE INPUT WORD AND 1 BYTE OF PHASE DATA PER 4-BYTE OUTPUT WORD
  - DATA TRANSFERS FROM COMMON MEMORY TO ON-BOARD MEMORY TRANSFER 1 UNUSED BYTE FOR EVERY BYTE OF USEFUL INPUT DATA
  - TRANSFERS FROM ON-BOARD MEMORY TO COMMON MEMORY TRANSFER 3 BYTES OF UNUSED DATA FOR EVERY BYTE OF RESULT DATA
- CORDIC WITH DATA PACKING
  - PACK INPUT DATA BEFORE CALLING MAP ROUTINE, 2 I VALUES AND 2 Q VALUES PER 4-BYTE INPUT WORD
  - TRANSFER DATA FROM COMMON MEMORY TO ON-BOARD MEMORY
  - CALL MAP ROUTINE
  - UNPACK DATA INSIDE MAP ROUTINE
  - EXECUTE CORDIC ALGORITHM
  - PACK RESULT DATA, 4 PHASE VALUES PER 4-BYTE OUTPUT WORD
  - TRANSFER DATA FROM ON-BOARD MEMORY TO COMMON MEMORY

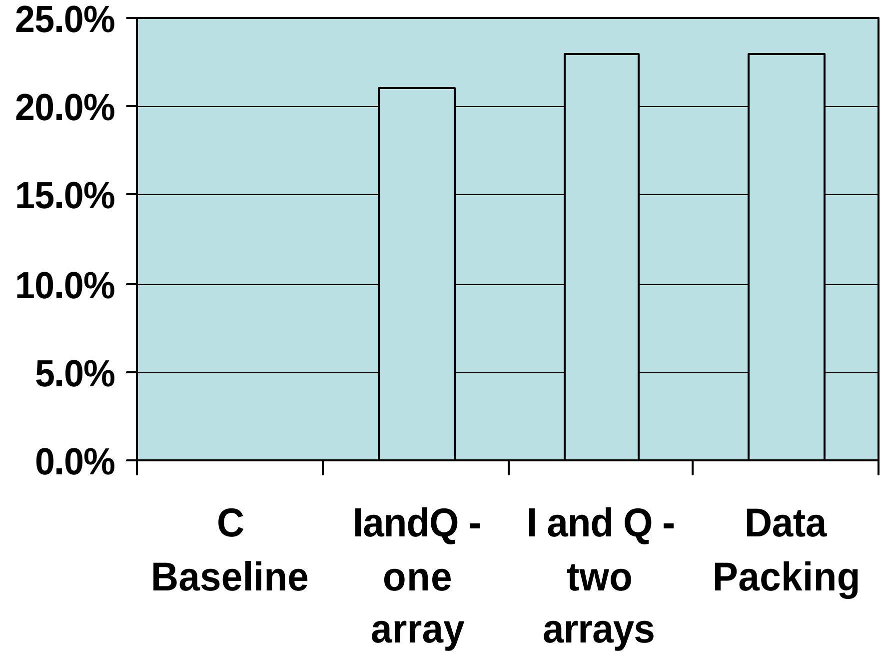
TEST RESULTS

EXECUTION TIME



190% INCREASE

FPGA SPACE USAGE



INSIGNIFICANT CHANGE

DECREASE IN MEMORY COPY TIME SIGNIFICANTLY LESS THAN TOTAL TIME REQUIRE TO PACK AND UNPACK INPUT AND OUTPUT DATA

MAP CODE GENERATED WITH CORE GENERATOR TOOL

	PRECISION (BITS)			ITERATIONS	INTERNAL LATENCY (CLOCKS)	CORE CIRCUIT AREA (SLICES)
	INPUTS	OUTPUT	INTERNAL			
CORE 1	8	9	16	11	17	475
CORE 2	8	9	32	11	17	777
CORE 3	8	9	48	30	36	2604

NAVAL POSTGRADUATE SCHOOL

TEST CASE RESULTS

Baylor University Engineering & Computer Science

	TOTAL CIRCUIT AREA (SLICES*)	TOTAL LATENCY (CLOCKS)	CORDIC EXECUTION TIME (SECONDS)
CORE 1	3260	68	7.3
CORE 2	3555	68	-
CORE 3	5450	81	7.5
C CODE	6710	112	7.4

*33,792 SLICES AVAILABLE

RESULTS ANALYSIS

- C CODE REQUIRED 2 TO 5 TIMES MORE CIRCUIT AREA
  - 6710 SLICES COMPARED TO 3555 FOR MOST SIMILAR IP CORE
  - APPROXIMATELY 2800 SLICES ADDED TO CORDIC CODE FOR SNAP PORT INTERFACE AND CONTROL
  - 3910 SLICES COMPARED TO 755 SLICES FOR MOST SIMILAR IP CORE
- CIRCUIT AREA LIMITS PROGRAM SIZE
- EXTRA FPGA AREA CAN BE USED FOR LOOP UNROLLING AND PARALLELIZATION TO INCREASE EXECUTION SPEED
- TOTAL EXECUTION TIME IS APPROXIMATELY THE SAME AT 7.4 SEC
- LATENCY FOR C CODE IS ABOUT TWICE THAT OF IP CORES
- LATENCY IS INSIGNIFICANT WITH LARGE DATA BLOCKS
  - Time per loop = (Latency – 1 + Number of Calculations) * 10 ns
  - (112 – 1 + 249984) * 10 ns = 2.5 msec/loop
- CORDIC EXECUTION TIME IS SMALL COMPARED TO TOTAL EXECUTION TIME OF 7.4 SEC
  - 256 loops * 2.5 msec = 0.64 seconds
- MEMORY TRANSFER TIME DOMINATES EXECUTION TIME
  - MEMORY TRANSFER REQUIRE 4 SECONDS AT PEAK TRANSFER RATE

COMPARISON OF MAP PROGRAMMING METHODS

- USE OF MAP C COMPILER
  - LITTLE SPECIAL KNOWLEDGE OF MAP HARDWARE REQUIRED
  - FLEXIBILITY -- PROGRAMMER IMPLEMENTS ALGORITHM AS DESIRED
  - LIMITED TO DATA TYPES RECOGNIZED BY COMPILER
  - LOWEST COST METHOD FOR APPLICATIONS WHERE IP CORES DO NOT EXIST
  - AVOIDS PURCHASE OR ROYALTY COST OF IP CORES WHEN THEY DO EXIST
- USE OF IP CORES
  - REDUCED DEVELOPMENT EFFORT
  - PREVIOUSLY VALIDATED (HOPEFULLY)
  - EASILY RECONFIGURED
  - LOWER LATENCY (PROBABLY NOT SIGNIFICANT)
  - MORE COMPACT CIRCUIT REALIZATION

NAVAL POSTGRADUATE SCHOOL

Baylor University Engineering & Computer Science

CONCLUSIONS

- SRC-6E COMPILER ALLOWS C PROGRAMMERS TO ACCELERATE PROGRAMS WITHOUT BEING CIRCUIT DESIGNERS
- PORTING CODE TO MAP REQUIRES BASIC KNOWLEDGE OF MAP HARDWARE
- MAP C COMPILER PRODUCES CODE THAT COMPARES WELL TO CUSTOM IP CORES
- DEVELOPMENT ENVIRONMENT HAS CAPABILITY TO INTEGRATE IP CORES OR CUSTOM CIRCUIT DESIGNS
- OVERALL PERFORMANCE CAN BE LIMITED BY MEMORY TRANSFER TIME BETWEEN COMMON MEMORY AND ON-BOARD MEMORY
- USE OF LARGE DATA SETS AMORTIZES MAP PIPELINE LATENCY ACROSS MANY CALCULATIONS
- APPLICATIONS PERFORMING A LARGE NUMBER OF CALCULATIONS ON EACH DATA SET DERIVE THE LARGEST PERFORMANCE BOOST FROM USING THE MAP